AMENDMENTS TO THE SPECIFICATION:

On page 1 of the specification, replace the paragraph beginning on line18 and ending on line 23 with the following paragraph:

When creating a computer application, the developer typically must chose a particular environment, or platform on which the application will ultimately be executed. For example, when writing an application, the developer must choose either the Microsoft Windows ® platform, the ~~Linux~~ LINUX™ platform, or some other platform. As a result of this choice, the program developer may have different options available for writing the application.

On page 3 of the specification, replace the paragraph beginning on line 16 and ending on line 25 with the following paragraph:

In accordance with other aspects, the present invention relates to a method of displaying data according to an appropriate style ~~comprising~~. The method involves receiving a request to display one or more data items and then locating the appropriate visual style, wherein the style is independently defined from the data items. Next the method generates a visual tree using the data items and the appropriate style and binds properties in the visual tree to properties of the data items. Last the method renders the display based on the visual tree. In an embodiment, the process involves declaring the data items using data objects and automatically updating the visual tree in response to a change to a relevant data item. The change to a relevant data item may involve an edition, addition or deletion of a relevant data item.

On page 11 of the specification, replace the paragraph beginning on line 10 and ending on line 16 with the following paragraph:

In an embodiment, the application module 302 has a data binding section 312 that, during operation, causes data content ~~items 309~~313, i.e., properties of the data objects 306, to be ultimately bound to the UI elements or properties of a style definition such as definition 315. The data binding section 312 relates to the declarative program statement(s) that associates one or more data objects to a data style. Such an association may be made by explicitly identifying the data types and the data style for that type, or by providing a style selector call, or by providing a default style, or by some other method.

On page 11 of the specification, replace the paragraph beginning on line 17 and ending on line 25 with the following paragraph:

Application module 302 operates on platform 314. Platform 314 relates to the framework or application programming interface (API) that provides the necessary communication between the application module 302 and the operating system of the computer, such as computer 100 shown in Fig. 1. As such, the platform 314 provides the intermediate functions and services to allow the application module 302 to ultimately display a list of items, such as objects 306, on a display 316. Although not shown, the operating system and its necessary operations occur between platform 314 and display 316. In a particular embodiment of the invention, the platform 314 is the Microsoft Windows ® platform developed by Microsoft Corporation.

On page 13 of the specification, replace the paragraph beginning on line 3 and ending on line 24 with the following paragraph:

A listing of some exemplary markup language, e.g., XAML code is provided in Fig. 4 to illustrate the defining of a style, such as those defined in style definitions 310 for use by the tree assembler module 318 (Fig. 3). As may be appreciated, the code snippet 400 is only one example of the way in which the concepts described herein may be implemented and should not be considered limiting to the shown syntax. Line 408 represents the declaration or definition of a new style named "EmployeeStyle". The name here is merely an example that continues the example shown and described above in conjunction with Fig. 1 wherein a number of employee objects may be listed. Following the definition of the style name, the markup describes the visual tree in the next seven lines. The visual tree consists of a dock panel that contains three text controls 410, 412 and 414. The visual tree essentially states that when an employee is being displayed, three properties of the employee are to be shown as text (via the Text controls), using layout rules supplied by the surrounding DockPanel. The first control 410 binds the data from the name field of the employee object to be displayed first, such as in the first column. As may be appreciated other details may be required to make this example work, such as adding width dimensions for the columns to each text control, etc. Text control 412 binds the data from the address field of the employee object to be displayed second, such as in the second column. Next, text control 414 binds the data from the employee ID field of the employee object to be displayed third, such as in the third column. The style may be applied to all employee objects at runtime such that the style does not have to be associated with the data until runtime. Further,

this one style definition can be applied to all employee objects such that the style does not have to be repeated when the employee objects are created.

On page 20 of the specification, replace the paragraph beginning on line 13 and ending on line 30 with the following paragraph:

For instance, during the execution of the application, a new object may be added to the object collection, such as object collection ~~306~~ 307 shown in Fig. 3, e.g., by use of an add item control such as control 116 shown Fig. 1. In such a case, the tree assembler module, such as tree assembler module 318 is notified of this change. The notification may be an active notification calling to the tree assembler module, or alternatively, the notification may result from the tree assembler module "listening" for such an event. Upon being notified of the change, the tree assembler module may determine whether the insertion is relevant. That is, the tree assembler module may determine whether the change will actually change the current display. In some cases, the tree assembler module may only generate user interface elements to fill one screen or display at a time and thus if the change to the object collection results in an insertion into an "off-screen" area then the tree assembler module may determine that the change is not relevant for the current display and do nothing. If however, the tree assembler module determines that the change is relevant, then the tree assembler module generates or changes the visual tree to include the new objects data items as discussed above and inserts the new user interface elements in the correct place within the visual tree. The new data object will get styled as described above and then the visual tree will be passed to the rendering engine to be displayed as discussed above.